DOI: 10.1111/cgf.13117

EUROGRAPHICS 2017 / L. Barthe and B. Benes
(Guest Editors)

Volume 36 (2017), Number 2

Texture Stationarization: Turning Photos into Tileable Textures

Joep Moritz Stuart James Tom S.F. Haines Tobias Ritschel Tim Weyrich

University College London

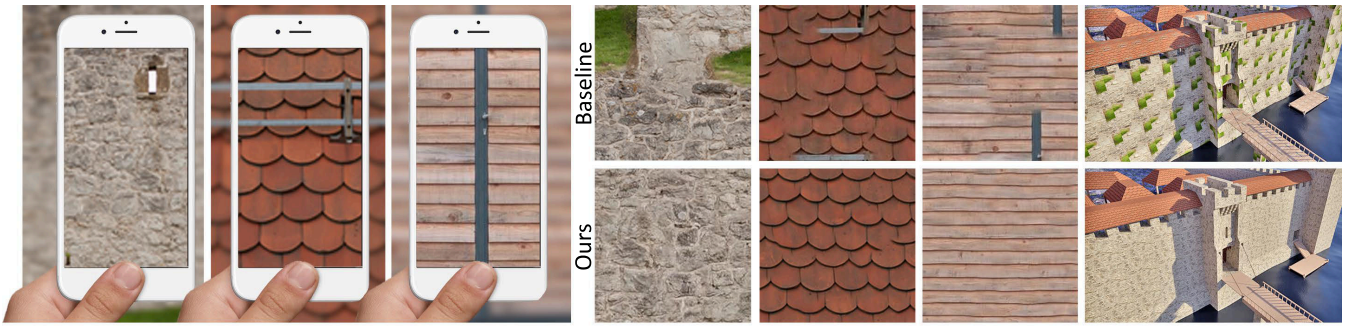

Figure 1: *Casually captured photographs* (left) *may contain a texture* (middle) *that a user would like to place in, e.g., a computer game* (right). *A baseline approach (PatchMatch [BSFG09] with toroidal boundaries) will produce noticeable repetitions in such input* (top row). *We include a measure of stationary into texture synthesis* (bottom row) *to generate tiling textures ready to be plugged into any renderer.*

Abstract

Texture synthesis has grown into a mature field in computer graphics, allowing the synthesis of naturalistic textures and images from photographic exemplars. Surprisingly little work, however, has been dedicated to synthesizing tileable textures, that is, textures that when laid out in a regular grid of tiles form a homogeneous appearance suitable for use in memory-sensitive real-time graphics applications. One of the key challenges in doing so is that most natural input exemplars exhibit uneven spatial variations that, when tiled, show as repetitive patterns. We propose an approach to synthesize tileable textures while enforcing stationarity properties that effectively mask repetitions while maintaining the unique characteristics of the exemplar. We explore a number of alternative measures for texture stationarity and show how each measure can be integrated into a standard texture synthesis method (PatchMatch) to enforce stationarity at user-controlled scales. We demonstrate the efficacy of our approach using a database of 118 exemplar images, both from publicly available sources as well as new ones captured under uncontrolled conditions, and we quantitatively analyze alternative stationarity measures for their robustness across many test runs using different random seeds. In conclusion, we suggest a novel synthesis approach that employs local histogram matching to reliably turn input photographs of natural surfaces into tiles well suited for artifact-free tiling.

Categories and Subject Descriptors (according to ACM CCS): I.4.8 [Image processing and Computer Vision]: Scene Analysis—Shape

1 Introduction

Textures, that is, spatially-varying appearance, are a visually important aspect of real-world scenes. A variety of methods emerged in computer graphics early on to efficiently mimic the richness of real-world textures: classic texture mapping [BN76], still the standard in many interactive applications, procedural textures [Per85] and synthesis-by-example [HB95, EL99], which allow computational creation of texture across arbitrarily large domains, as well as acquisition using advanced measurement devices [DvGNK99], and last but not least user-guided texture generation [Hae90].

Due to their small memory footprint, rectangular textures that lend themselves to a tiling of the plane play a central role in real-time computer graphics. Ideally, such a texture tile exhibits toroidal continuity, that is, top and bottom, and left and right edge, respectively, seamlessly fit together. That alone, however, is not sufficient to create the illusion of a continuous surface texture when tiled. Depending on the texture content, individual features may visually "stick out" as repeating patterns under tiling. In practice, this effect is mitigated by skilled artists who carefully warp and blend original texture content to create a continuously tiling base image.

Surprisingly little work, however, has been dedicated to *synthesizing* tileable textures computationally, despite the mature body of work on texture synthesis in general. Automated generation of tileable textures from unconstrained image materials would be invaluable in applications that employ user-provided content, such as, casual massive multi-player online or augmented-reality games, where one cannot assume users to be experts in controlled capture and professional post-processing.

We address this challenge by devising an approach to convert any given texture-like image into a texture that only exhibits a user-prescribed maximal amount of perceived repetitiveness when tiled. To this end, we first introduce different measures of perceived stationarity . A signal is called stationary if, according to some measure, it is perceived similar across a domain. We then suggest a texture re-synthesis that explicitly enforces a prescribed amount of perceived stationarity.

A perfectly stationary texture would be a uniform color that tiles perfectly (no visible repetition), defeating the purpose of texturing. Strongly non-stationary textures tend to be visually rich, but appear repetitive when tiled. In between lies a continuum that we explore. Our method allows synthesis of tileable textures with different degrees of stationarity, allowing a user to pick the version most suited for their application. Beyond that, we demonstrate a perceptually-motivated approach to linearize perceived stationarity to produce a texture sequence that shows an even progression of perceived stationarity, allowing for an even more accurate selection by the user. In summary, our main contributions are:

- a set of measures for perceived texture stationarity,
- an algorithm to enforce a desired stationarity for a texture,
- perceptually-motivated linearization of stationarity.

After reviewing previous work in Sec. 2, we propose our measure of stationarity in Sec. 3 which is then used for "stationarization" in Sec. 4; we show results of our approach in Sec. 5.

2 Previous Work

After recalling fundamentals of texture perception, in particular spatial grouping, we will relate our approach to existing work on texture synthesis and extraction.

Texture perception The importance of texture on perception has been acknowledged long before algorithms to analyze textures had been proposed [Att54]. We will here recall the most relevant aspects while referring the reader to the survey by Rosenholz [Ros14].

A question of particular importance to our objective is the one of *texture segmentation* [Jul62], i.e., a formal way of separating regions which are perceived to have different texture. In the case where multiple texture segments are present in an image, the image requires stationarization, otherwise the result will look repetitive when tiled.

Different hypotheses exist about how the human visual system performs segmentation, but most agree that the image is first converted into a feature representation and then grouped, such that areas with similar features appear as one segment. The simplest descriptor is the 1st- and 2nd-order moments [Jul62] (mean, variance), while evidence exists that higher orders (skewness, kurtosis, etc.) contribute little to the perception of stationarity. We experiment with moments of different order and indicate where they have an impact on visual quality.

However, some textures have identical 1st- and 2nd-order moments yet their primal and gradient derivatives still have a clearly perceivable segmentation [VB78], historically formalized in terms of Buffon needles [Jul65]. To address this, we can think about arbitrary spatial aggregations of RGB values into features, where the mean and variance of gradients are special cases [Cae85].

Extending derivatives, texture can be considered the combination of bases, such as steerable filters [FA91], which quantify changes of different frequency and orientation. Or alternatively as a combination of responses to filters, commonly formalized as 'Textons', which quantify the response. Due to the indirect nature of textons, we experiment with the former, steerable filters.

The perception of texture has been explored in statistics, geology and the environmental sciences, with a comprehensive overview by Taylor et al. [TEN14]. Common approaches utilize wavelets, as in [TEN14], extending prior work on univariate time series to the spatial domain; alternatively there exist Bayesian methods [Fue05]. The work in these fields is similarly grounded in 1st and 2nd moments [PR69]. It should be noted that these approaches focus on images containing a unique texture, e.g., carpet, as opposed to casually acquired images that contain multiple textures. This results in the proposed measures focusing on macro variations within the texture, in contrast to the broader discrepancies e.g., stains, rusting, or cracked paint.

Following these ideas, we devise a method that seeks to achieve the opposite of segmentation methods, while being guided by statistics literature: by enforcing stationarity in the perceptual feature space to drive synthesis of a repeatable texture we explicitly avoid generating textures with elements that appear repetitive when tiled.

Texture synthesis Synthesis of images is a traditional computer graphics problem, with multiple avenues in the literature. We choose to focus on the problem of preventing perceived repetition within a synthesized image, and focus on relevant methods to this end.

Texture synthesis literature has focused on offline methods, where deep visual analysis or computation can be performed. Early approaches focused on procedurally generated textures [Per85] at high computation cost and complex parameters. Avoiding the complex parameters, Efros et al. [HB95, EL99, EF01] stitched together pixels or patches from a source image to synthesize non-repeating textures. These approaches avoided repetition, by adding infinite amounts of variation on demand, but are not inherently tileable.

Given an arbitrary input exemplar texture, synthesizing a new one can be seen as an optimization problem [KEBK05, WSI04]: each neighborhood in the new image should be as similar as possible to a neighborhood in the source image. Optimization-based methods produce images that have the same feature statistics as the input according to different feature spaces, but it is less clear what happens if the input does not have homogeneous statistics, i.e., it has more than one segment. Typically, the stationarity requirement is not explicitly addressed, as the input is assumed to be stationary. For moderate non-stationary input, these methods do not necessarily fail, as often the underlying stochastic optimization is able to find a local minimum that is still tileable. In many cases, and in the presence

of more noticeable non-stationarity in the input, however, a tiling of the synthesized texture leads to a clearly visible repetition. This problem persists, even when toroidal boundary conditions (see fig. 1), a prerequisite for tileability, are included in the optimization.

PatchMatch [BSFG09] is a fast algorithm to find (local) minima of such an optimization problem. Our approach will build on this method, by extending it with toroidal boundary conditions and stationarity control.

A recent trend in texture synthesis is to use convolutional neural networks, first used in style transfer [GEB15, AAL16, JAFF16]. In Aittala et al. [AAL16] a histogram prior is used to encourage stationary solutions when using a neural network to optimize for a texture. Their results show stationary textures, and the prior assures avoiding non-stationary results. As with the statistics literature, Aittala evaluated on already largely homogeneous input textures.

Another approach to hiding repetition during texturing is to use real-time texture synthesis [LH05, LL12]. Vanhoey et al. [VSLD13] use offline segmentation to propose variation in the synthesized texture at runtime. The reliance of such approaches on custom GPU code and large runtime resources makes them less attractive to adoption. Therefore, we focus on achieving visually appealing offline results that can be used in any standard rendering environment.

Stationary Texture Synthesis Typically, in example-based texture synthesis, inputs are assumed broadly stationary [HB95, EF01, AAL16], thus lending themselves to stationary output.

Dai et al. introduced a measure of "synthesizability" [DRV14] that is learned from a set of over 20,000 annotated textures. This measure can be used to crop the most synthesizable rectangle from a texture for synthesis purposes. However, the useful source area is not always a rectangle, and the "offending" elements might be scattered, such that a single rectangle is a poor fit. Even from a broadly stationary input, synthesis can fail and produce non-stationary output that cannot be tiled.

Common deviations from stationarity occur due to staining [YY14] and/or weathering in natural textures [BKCO16]. Reducing weathering also implicitly increases stationarity.

The most acknowledged approach to non-repetitive texturing with limited resources is Wang tiles [CSHD03, LD06, XM10, YBY*13]. As Wang tiles are tiled along each same-colored edge, they need to solve a similar problem to ours: making a texture tileable. However, these methods, again, start from a stationary input and use graph cuts to retain stationarity across all same-colored corners or edges. While Wang tiles are attractive from a theoretical point of view, and can by design produce infinite, non-regular layouts, they have not yet found wide-spread use in interactive applications such as computer games.

Texture extraction Less attention has been devoted to extracting the dominant texture from complex images. While our approach is fully automatic and not explicitly for a single texture, for completeness we include a review of these techniques here.

Building on texture perception, texture segmentation can be used to find the dominant texture in an exemplar [LDR09, WH13] by identifying regions of the input that are themselves stationary. Alternatively Lu et al. [LDR09] and Lockerman et al. [LSA*16] segment images in a multi-scale fashion to generate a texture palette that can be used for texture painting. They also show an application where they pick the largest segment and synthesize using that. But, the approach is unable to control the amount of stationarity of the resulting texture, and therefore is ill-equipped to source multiple segments while maintaining stationarity.

An intuitive approach is to require the user to select the desired region to be synthesized. For example Eisenacher et al. [ELS08] use a selected area to build a small 3D model. Using the 3D model it can correct for perspective and use the photographs to synthesize a texture. It can also use the 3D model to synthesize back into the image; Eisenacher did not consider the tileability of the texture.

3 Perception of Visual Stationarity

Perception of stationarity has not been explicitly accounted for in texture synthesis. We suggest several plausible measures, before making our recommendations on how to enforce the stationarity of an exemplar. For our purpose, we not only need to passively measure stationarity, but also need a practical way to enforce it, which will be discussed in Sec. 4.

To account for stationarity, we need to find a notion of stationarity flexible enough to compare different theories of texture perception, that is computationally reasonable, and can be included in an optimization. To this end, we express the degree to which a signal $f(\boldsymbol{x})$ deviates from stationarity across a domain D according to the measure $\mathcal{S}$ as the summed differences between a feature-space representation of the signal at any two locations in the domain:

$$s(f) = \int_{D^2} \mathcal{S}(f)(\boldsymbol{x}_1) \ominus \mathcal{S}(f)(\boldsymbol{x}_2)\, \mathrm{d}\boldsymbol{x}_1 \mathrm{d}\boldsymbol{x}_2\,, \tag{1}$$

where $\ominus$ is a suitable difference between perceptual feature representations. In other words, as $s(f)$ approaches zero, the texture f is perceived the same everywhere. Note, that $\mathcal{S}$ is a functor, mapping the entire function to a new (feature) space, and not a function of the signal value at any single location. This allows us to take the entire spatial neighborhood into consideration, for instance to build a histogram. We note our stationarity is similar to saliency [IKN*98], which also identifies locations that are different to other locations according to a perceptual measure.

We will here consider 2D textures, so, without loss of generality, D is the unit square. Note that stationarity is only used to assess differences within the texture, not in respect to any exemplar. The following measures and projections are performed over scalar values and are done independently in a de-correlated perceptual color space — we use CIE L*a*b*. Additionally, while there are a variety of improved measures for distances between two colors in this color space, we opt for the standard of Euclidean distance.

3.1 Moments

The easiest notion of stationarity is to consider the vector of up-to-p-th-order-weighted local moments, as in

$$\mathcal{S}_{\mathrm{mom}}(f)_p(\boldsymbol{x}) \in D \to \mathbb{R} = \int_D w_{\mathrm{d}}(\boldsymbol{x}, \boldsymbol{y}) f(\boldsymbol{y})^p \mathrm{d}\boldsymbol{y}\,, \tag{2}$$

where $w_{\mathrm{d}}(\boldsymbol{x}, \boldsymbol{y})$ is a spatial weight function such as the unit Gaussian. The first moment $p = 1$ is the mean, the second at $p = 2$ the variance, etc. The earliest theories of texture segmentation [Jul62] have used moments as descriptors, before counter-examples have indicated

the human visual system (HVS) might do more complex operations. Our results also indicate that enforcing stationarity based only on a notion of moments is at the expense of increasing the texture cost, i.e., producing results that do not look like the exemplar. As a distance operator $\ominus$ we use squared difference.

3.2 Histogram

A more complete description of local statistics are weighted local histograms with n bins:

$$\mathcal{S}_{\text{histo}}(f)(\boldsymbol{x}) \in D \to \mathbb{R}^n = \int_D w_{\text{d}}(\boldsymbol{x},\boldsymbol{y}) w_{\text{r},i}(f(\boldsymbol{x}))\,\text{d}\boldsymbol{y}\,, \qquad (3)$$

where w_{d} is the domain weighting function, a unit Gaussian with standard deviation σ, a key control parameter to explore and w_{r} is the range weighting function of bin i, (a Parzen window) typically a narrow Gaussian or a box. As a distance operator $\ominus$ we use earth mover distance.

In texture synthesis, others have matched the histogram of the (stationary) input exemplar texture to the current output solution [KFCO*07, KNL*15]. In our approach, we match local histograms at different positions within the output exemplar produced, consuming a non-stationary input.

3.3 Steerable Filters

One popular interpretation of texture perception is that it is merely a by-product of low-level to mid-level vision where the complex cortical areas V1 and V2 convolve the retinal image with a bank of filters. A simple example of such a bank are steerable filters [FA91], that select different orientations and magnitudes. In practice, the definition is equivalent to the histogram, except that the binning is done on each steerable response channel, instead of on each color channel. The response of asteerable filter with orientation o and magnitude m, is then

$$\mathcal{S}_{\text{steer}}(f)(\boldsymbol{x}) \in D \to \mathbb{R}^n = \int_D w_{\text{d}}(\boldsymbol{x},\boldsymbol{y}) w_{\text{r},i}(F_{o,m} * f(\boldsymbol{x}))\,\text{d}\boldsymbol{y}\,, \qquad (4)$$

where the definitions from above apply, $*$ is convolution and $F_{o,m}$ is the respective steerable filter kernel for orientation o and magnitude m [FA91]. We use the freely available implementation of steerable pyramids [Sim15] by Simoncelli, with the default parameters for o (4) and m (1), resulting in 12 dimensions in total (4 for each color).

4 Texture Stationarization

Given a suitable measure for perceived non-stationarity from the previous section, we now wish to synthesize (tileable) textures while aiming at minimizing this measure.

Similar to Heeger and Bergen's method [HB95], we use an iterative scheme where synthesis steps interleave with steps where we re-project the data onto a desired manifold. In contrast, however, we do not enforce the global statistics of the exemplar (which would only make sense if the input was assumed stationary); instead, our re-projection step aims at minimizing a stationarity cost function based on a measure from Sec. 3. This leaves room for combination with an existing texture synthesis method, similar to Aittala et al.'s work that uses a moments-based prior [AAL16] to steer the gradient-descent of a CNN-based texture generator.

In our case, we chose to use the widely successful PatchMatch algorithm [BSFG09]. Its hierarchical, iterative scheme lends itself naturally to repeated re-projection of the intermediate solution (Sec. 4.2). Accordingly, we extend this algorithm by adding a new stationarity cost (Sec. 4.1) to the original texture cost, as well as toroidal boundary conditions to ensure tileability of the result. We re-project the current solution onto the space of stationary textures after each iteration of PatchMatch (Sec. 4.3).

4.1 Cost

Given a texture image f, we measure, and optimize for two cost functions separately: c_{text} and c_{stat}. The texture cost c_{text} expresses that the result texture f should locally look like some neighborhood in the exemplar e [WSI04]. Formally, this is the integral of the minimal difference of all neighborhoods in the result and the exemplar,

$$c_{\text{text}}(f) = \int_D \min_{\boldsymbol{y}\in D} \left\{ \int_P \|f(\boldsymbol{x}-z) - e(\boldsymbol{y}-z)\|_2^2 \text{d}z \right\} \text{d}\boldsymbol{x}\,, \qquad (5)$$

where P is the domain of a texture patch expressing a local neighborhood (with toroidal connectivity, see below). The size of P in our implementation is always 11×11 pixels at a typical texture image resolution of 256×256, but we are not limited to this size. The stationarity cost $c_{\text{stat}}(f)$ is simply s from Eq. 1, $c_{\text{stat}}(f) = s(f)$, and depends on the given measure $\mathcal{S}$.

4.2 Optimization

The algorithmic outline closely follows PatchMatch as originally formulated [BSFG09].

PatchMatch uses a multi-scale approach. The top level of our scale pyramid starts at 64×64 pixels, or 1/4 the size of the final image. We perform 30 iterations at this level, letting the algorithm do most of the work here. At subsequent levels convergence is reached within 2 or 3 iterations, making mostly minor adjustments to account for up-scaling of the image. We scale up by a factor of 1.3 [KNL*15].

We also include two typical improvements found in the recent literature, making adjustments for our setup. The first is the spatial uniformity term of Kaspar et al. [KNL*15] that ensures the resulting texture uniformly samples from the exemplar. As the size of our high-resolution photos is much greater than the size of the resulting tileable texture, we need to adjust the spatial uniformity term. We change the formula for ω_{best}, which represents the ideal number of times an exemplar pixel is used in the synthesized result. Our new formulation allows pixels to always be used at least once:

$$\omega_{\text{best}} = \max(1, \frac{|T|}{|S|}) N^2 \qquad (6)$$

The second improvement is a global histogram matching between the resulting texture and the input exemplar similar to Kopf et al. [KFCO*07]. Unlike their algorithm, our histogram enforcement is not integrated into the "voting" phase. Instead, we simply match the histogram of the synthesized texture to a target histogram, at every iteration. This target histogram is created by blending the histogram of the exemplar with the histogram of the synthesized texture, using a given parameter α, and by interpolating along the x-axis of the CDF [MZD05]. This allows us to smoothly fade out the effect over multiple iterations, instead of turning it off abruptly like [KNL*15].

The full approach is outlined in Algorithm 1. Notice that we apply PatchMatch last, allowing the original pixel values of the source

image to produce the final image, unaltered by histogram matching or stationarization.

Input : Exemplar texture e
Output : New texture f
$f_0 := \texttt{randomScramble}(e_0)$
for *all pyramid levels* j **do**
  $f_j := \texttt{upsample}(f_{j-1})$
  for $i = 0$ *to* $n_{\max}$ **do**
    $f_j := \texttt{matchHistogram}(e_j, \alpha)$
    $f_j := \texttt{project}_{\mathcal{S}}(f_j)$
    $f_j := \texttt{patchMatch}(e_j, f_j)$
  end
end

Algorithm 1: Pseudo-code for synthesis of stationary textures.

Toroidal boundary conditions Traditional texture synthesis algorithms, including PatchMatch, do not produce tileable textures. We make the following modifications to PatchMatch to ensure circular boundaries:

- Any time a neighborhood is accessed in the synthesized image, it wraps around as needed. This affects the implementation of the original algorithm's distance function and its blending function used in the "voting" phase.
- At the edge of the synthesized image, the propagation step passes good matches on to neighbors on the other side.
- PatchMatch's nearest-neighbor field is defined for every pixel of the synthesized texture, no longer leaving a boundary undefined.

4.3 Stationarity Projection

At this point, the current solution is enforced to become stationary according to one of the measures $\mathcal{S}$ above. This can be seen as iterative optimization, where PatchMatch performs a step of optimization, and we perform a projection onto the manifold of stationary textures.

The `project` operation is implemented in different ways for different $\mathcal{S}$. This projection is not generally a linear operation. Previous work uses linear operations based on the first-order gradients of the cost [AAL16] or loss [JAFF16]. Our approach immediately enforces stationarity each iteration.

All projections occur independently in every color channel. We will now discuss the projections for different notions of stationarity.

Histogram The histogram measure $\texttt{project}_{\text{histo}}$ is a generalization of the moments to arbitrary distributions. First, cumulative histograms are built for every pixel from its neighborhood, using Gaussian-neighborhood weighting. We then match each local histogram to the global histogram as follows: the value of each pixel is updated so that $c' = G(F(c))^{-1}$, with F the cumulative histogram for the pixel, and G the cumulative histogram of the entire image.

An approximation and optimization of this local histogram matching starts by dividing the entire image into overlapping blocks of the neighborhood's size, with their centers spaced by k pixels, with $k = 2$ at the top of the pyramid, increasing to $k = 5$ toward the bottom. For each block, a weighted histogram is built, as before.

All of the pixels in each block are then adjusted by matching the block's histogram to the global histogram. The resulting, overlapping blocks are merged to form a new image by blending them together, using the same Gaussian weights. For the small k used we found the results virtually indistinguishable from dense local histogram matching, while significantly reducing run time.

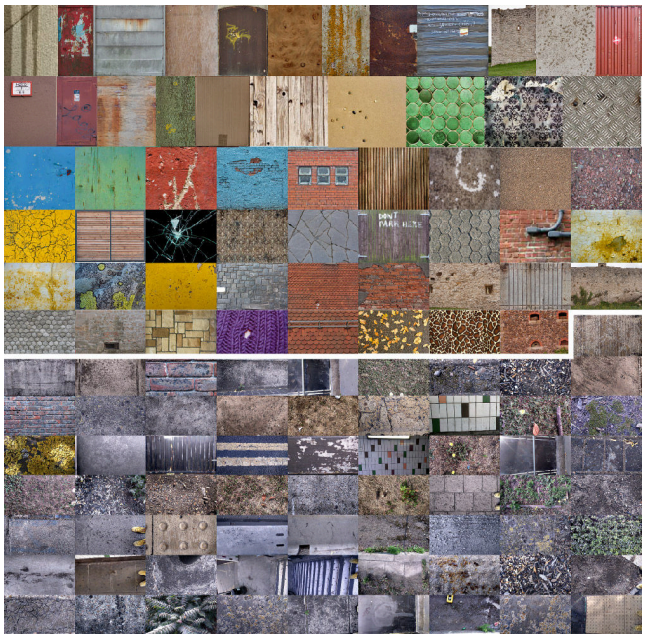

Figure 2: *Our data set of* 118 *non-stationary images. The upper part is from previous papers and the lower is new captures.*

Moments The same machinery that is used to project (complex) general histograms can be used with histograms described only by their (simple) moments. This requires converting back the moments into histograms. This is done by fitting a Gaussian distribution for second-order, or Pearson distribution for third and fourth (using the closed-form solution of Johnson et al. [JK70]), to have the correct moments, and by sampling them into discrete histogram bins.

Steerable filters Steerable filters produce an (over-complete) high-dimensional descriptor C for every pixel. We again build histograms for each dimension of these descriptors, followed by a local matching of every pixel to its surrounding histogram as explained for the histogram measure $\texttt{project}_{\text{histo}}$. The result is a new over-complete descriptor vector C'. Finally, we need to find an image that would result in C' when applying the filter bank. While several advanced methods to solve this inverse problem exist [SWRC06], we opted for the simplest one that just interprets C' as linear weights for a convex combination of basis patches to compute the result image, as done by Heeger and Bergen [HB95].

4.4 Perceptually-Motivated Stationarity Control

The main parameter of our approach is the size σ of the spatial weighting kernel: if σ is chosen large, the stationarization considers large areas of the image; if it is small, it will attempt to make small details look similar across the image. How much a result visually changes in response to changes in the kernel size σ, however, strongly depends on the content. To provide the user with a convenient way to control the effect, we initially sample the spatial kernel size domain using n different values $\sigma_1, \ldots, \sigma_n$ (Fig. 10).

In our experience, such a simple palette of gradated options is excellently suited to have a user select a suitable window size. To further quantify the effect of window size on perceived stationarity, and to find a more fine-tuned window size, we propose a texture-dependent remapping of window sizes into a perceptually-motivated space. To that end, we determine the difference in appearance between two results with consecutive window sizes σ_j and σ_{j+1}, $d_j = \|\sigma_j, \sigma_{j+1}\|_{\mathrm{M}}$, using a suitable difference measure M. Initially we experimented with established perceptual metrics (e.g., SSIM), but found them too sensitive to changes of feature locations. Instead, we settled on the earth mover's distance between histograms, as it does not depend on the location of features. Empirically we verified that is correlates well with perceived difference.

By inverting the mapping $\sigma_i \mapsto d_i$, we define an appearance-normalized space of window sizes. Re-synthesizing a texture for window sizes regularly distributed in that space, obtains a perceptually uniform progression of textures.

5 Results

After defining our data set in Sec. 5.1 and the procedure of instrumentation in Sec. 5.2, we report the main quantitative analysis in Sec. 5.3 and qualitative results in Sec. 5.4, including comparisons to other works, before concluding with applications in Sec. 5.5.

5.1 Dataset

Addressing the challenge of dealing with non-stationary images we contribute a new data set. The images contain textures that are both non-stationary and subject to the diverse challenges of casual images, such as slight perspective and shading. It consists of 118 images, visualized in Fig. 2. It contains *(i)* 16 textures from previous papers ([LDR09, BKCO16]); *(ii)* 38 publicly available textures and *(iii)* 64 difficult exemplars we acquired in an urban area on a cloudy day with a mobile phone camera. The data set is publicly available at `http://bit.ly/TexStat`.

5.2 Procedure

We will first introduce our protocol for comparing synthesized textures, which are stochastic and depend on random seeds, i.e., they are not point measures but random variables.

Protocol Due to the stochastic nature of PatchMatch, our approach — as well as many other example-based synthesis algorithms — can sometimes produce results of varying quality. One seed might be lucky and produce a stationary result without explicitly enforcing it such as we suggest. In fact, this can even be the majority of cases for inputs that are close to stationary with only a few deviations from stationarity. However, there is a minority of cases, where problems persist. We will later formally study how large this group is in our data set and how likely they are to fail to produce a stationary outcome. Our aim is to provide guarantees for avoiding them.

When comparing algorithmic variants based on (non-)toroidal PatchMatch, we make sure to use identical seeds. We found this essential to obtain more stable (less random) comparisons, while at the same time giving the baseline (toroidal PatchMatch) a competitive advantage: we first find the seed (out of 10) that produces the most stationary (lowest c_{stat}) result with toroidal PatchMatch, and then use that seed for all variants of our approach. We verified that, with inferior seeds, our stationarizing approaches would still lead to similar results while the baseline often performs worse.

We include a formal analysis, with random seeds, demonstrating our approach's stability, while other approaches produce outliers in terms of stationarity. Stationarity is best seen for an exemplar when either repeating it (in this paper in a 2×2 layout, or a perspective visualization that shows a "stress test" where the texture is used to fill the entire plane, at many different levels of scale.

Baselines We will compare against many alternatives in this paper, and in the supplemental material; they are: *(i)* directly use the input; *(ii)* run PatchMatch, i.e., without optimizing for stationarity and without toroidal boundary conditions; *(iii)* same as before, but with toroidal boundary conditions; *(iv)* the GIMP functionality "make seamless", which shifts the image by half in horizontal and vertical direction before a feathered blend; *(v)* variants of our approach using different notions of stationarity, such as *a)* moments, *b)* local histograms, and *c)* steerable filters. All these baselines are contained in the supplemental material, and we only show examples here.

If not said otherwise, we refer to a result of "our approach" as to a result of *(v).b*: optimization for stationarity using local histogram matching. We found this combination to provide the best trade-off in computational effort, complexity and visual quality. In the same way, if not said otherwise, we will refer to the "baseline" as to method *(ii)*: optimization for texture cost, without stationarity enforcement but with toroidal boundary conditions.

5.3 Quantitative Analysis

The plot in Fig. 3 shows a quantitative analysis of our approach. Instead of reporting anecdotal observations of single exemplar instances, our data set will allow us to consider a large ensemble of textures in terms of their statistics. Additionally, another methodological novelty of this work is to look at the "stability" of results under different random seeds.

Costs First, Fig. 3.a shows that the sum of the stationarity and texture costs (labeled combined cost) is going down as a result of our approach. This indicates, that we produce textures both similar to the input exemplar but also with desired stationarity. After 25 iterations, the cost has converged, and we can look at the distribution of costs across the data set in Fig. 3.b–d. We see that our distribution of costs (blue) compares favorably to the baseline, PatchMatch with toroidal boundary conditions (orange). As this cost is a sum of the texture and the stationarity cost, we look into the two in isolation in Fig. 3.c and d. We see that for the baseline, the stationarity cost is high, while the texture cost is low. By employing our stationarity enforcement, we reduce the stationarity cost significantly, while only slightly compromising on texture cost.

Stability Having a robust notion of cost enables us to look into what happens when the seeds are changed. Methods that work well, should be stable, i.e., the resulting cost should not change when the seed changes. To this end, we repeat the above experiment, for $N = 10$ random seeds. This number of samples N is suitable to predict the population mean and variance, as the 0.95-confidence t-test interval widths are only 0.014 for our mean, 0.013 for baseline mean, 0.060 for our variance and, 0.056 for the baseline variance, i.e., smaller than a bin in the chosen visualization.

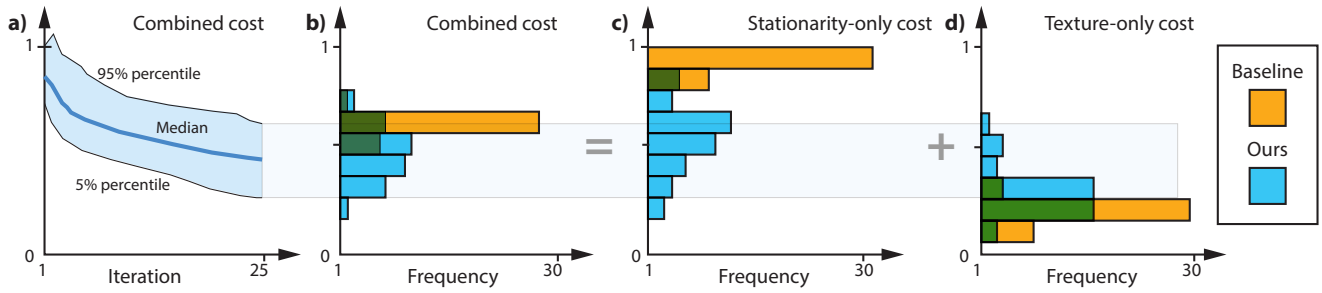

Figure 3: *Numerical analysis of cost optimization and cost distribution over our data set.* ***a)*** *Median* (thick line) *and 90-percentile confidence intervals* (thin lines) *of the total cost of all samples in our data set* (vertical) *as a function of iteration count* (horizontal). ***b)*** *Distribution of the total cost after the optimization has finished. We show the distribution as a transposed histogram, where the horizontal axis is frequency of occurrence in texture count and the vertical axis is cost, to match the axis of the first plot. The horizontal axis is the number of textures where the cost takes a certain value. We show both our approach* (blue) *and toroidal PatchMatch that does not account for stationarity in the synthesis* (orange). ***c:)*** *and* ***d:)*** *Stationarity cost and texture cost alone, again after optimizing both for the combined cost.*

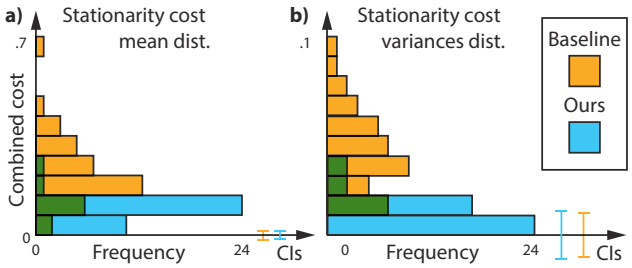

Figure 4: *Stability analysis: distribution of stationarity cost means* (left) *and variances* (right) *across the data set. Texture synthesis is repeated for each exemplar with ten different random seeds.* ***a)*** *Most of the textures have a lower mean cost* (blue) *than previous work* (orange) *indicating we perform better on average.* ***b)*** *At the same time, we see that the variance of the cost under random seeds is low, indicating, that we are not only better on average, but rarely worse. In contrast, previous work has a high variance, showing that it is not only worse on average, but also has a substantial number of cases where it fails. (0.95-confidence t-test intervals on the x-axis.)*

To understand the outcome, we now look a the mean and variance of the costs when seeds are randomized. The distribution of the mean and variance of the stationarity cost is seen in Fig. 4. Note, that these are not the mean and variances of the distributions we plot, but we plot a distribution of means and variances across an ensemble of statistics. In Fig. 4.a, we see that the stationarity cost is less, even when seeds change for our approach, an outcome in agreement with Fig. 3.c. More strikingly, we see that for the previous method (again PatchMatch with toroidal boundary conditions), many exemplars show a much higher variance. This indicates, that the previous approach can produce textures that are stationary, one has to be lucky: sometimes stationarity is a random by-product, sometimes not. In our case, variance is low, indicating it will always have a prescribed stationarity.

5.4 Qualitative Results

We now show qualitative visual results. The reader is encouraged to explore the interactive supplemental material, where we show results for our approach, as well as many alternative and competing approaches for the entire data set using different visualizations.

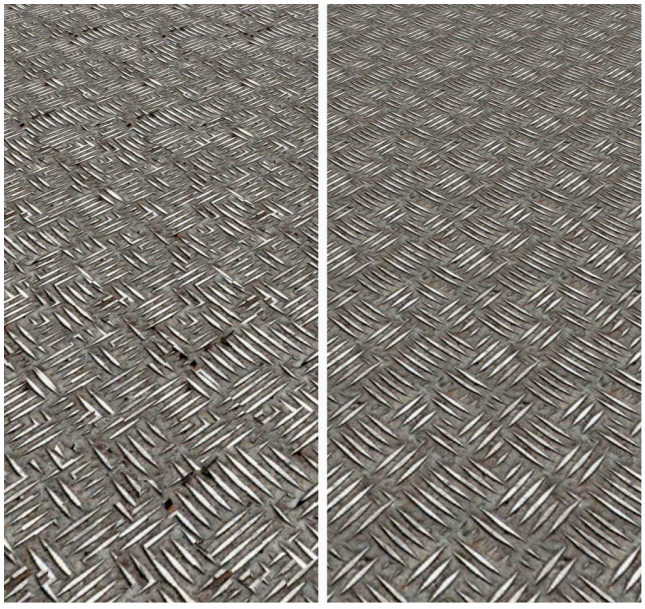

Figure 5: *Comparison between Wang tiles* (left) *and our approach* (right). *Both approaches successfully avoid obvious periodicity, but Wang tiles show a visually distinctive patch of rust repeatedly.*

Typical results of our approach are shown in Fig. 13.

Consideration of Stationarity For a visual comparison of stationarity we consider two different aspects — firstly the homogeneous nature of the generated texture and secondly the degree of preserved visual characteristics. The latter competes with the notion of stationarity, resulting in a trade-off between the two. In some cases the artist might prefer a homogeneous result, but usually they prefer to maintain characteristic details of the source texture.

Comparison of stationarity measures Applying these considerations of stationarity, Fig. 9 shows a comparison between the different methods for enforcing stationarity. The various synthesized 'brick' textures show how histogram matching is able to maintain both the structure and the high frequency information, that is lost through moments. Alternatively, "paint" demonstrates the advantage of the

averaging effects that hinder "brick", producing a very even texture. It is the responsibility of the artist to decide on the suitability of each image. These results are indicative across the data set, emphasizing that moment enforcement of stationarity is more likely to produce more uniform textures, while more complex textures can be generated using histogram matching or steerable filters.

Comparison to Wang tiles Fig. 5 compares our approach to Wang tiles [CSHD03] , generated with two colors on each edge to obtain 16 tiles. The results are broadly similar, but Wang tiles are unable to avoid non-stationary elements. Consequentially, while the patch of rust may not form a regular pattern its repetition is still noticeable. It would be possible to combine our approach with Wang tiles to obtain the benefits of both. Finally, our approach works as a transparent preprocess on the texture image, while Wang tiles require both access to the render code, as well as more memory and computational effort.

Comparison to Bellini et al. The different but related task of de-weathering an image has been addressed by Bellini et al. [BKCO16]. In Fig. 6, we applied both our baseline synthesis, as well as our proposed algorithm, to both input and output of their algorithm. In each scenario, our approach successfully stationarizes the texture, while it is evident that the output from [BKCO16] is largely de-weathered but not consistently stationary. Particularly of interest are Fig. 6's 3rd column, odd rows, where we produce a stationary, but not de-weathered, tileable texture from an original exemplar. Also note how using the two algorithms in tandem (3rd column, even rows) yields meaningful, i.e., de-weathered and stationarized results.

Comparison to Lu et al. We ran our implementation of [LDR09] against our data set to produce a segmentation of each photo. Their algorithm identifies one segment as the dominant one, which we use to generate a binary mask. We perform standard texture synthesis using PatchMatch with toroidal boundaries, with the mask identifying areas in the photo to use as input data. Areas outside of the mask are excluded entirely. For comparison we run the same synthesis algorithm, with our stationarity enforcement enabled. All results (Fig. 7) appear stationary, but those of Lu et al. lose detail, as it appears too aggressive in excluding salient features. Our approach preserves more larger-scaled details, while the segmentation greedily picks an "easy" homogeneous area from which to synthesize exclusively.

Limitations Failure cases of our approach are shown in Fig. 8. The first column of Fig. 8 shows how large scale structure is effectively removed by our algorithm, even though repetition of this structure would not look out-of-place. This is a particularly common problem with man-made objects. It can be avoided by choosing a large value for σ, but this would decrease stationarity by allowing smaller features to vary spatially. In the case of Fig. 8, the rust stains would re-appear. Future work could investigate methods to control stationarity separately at different levels of scale, or explicitly account for man-made structures similar to [KNL*15].

In a similar vein, the second column illustrates a problem with tiny salient features: the little pebbles are too small to be identified and removed by the algorithm. The third column of Fig. 8 illustrates our cognitive ability to pick out certain shapes particularly well (the small black circle), while the algorithm does not see anything out of the ordinary. Further work may detect such salient features.

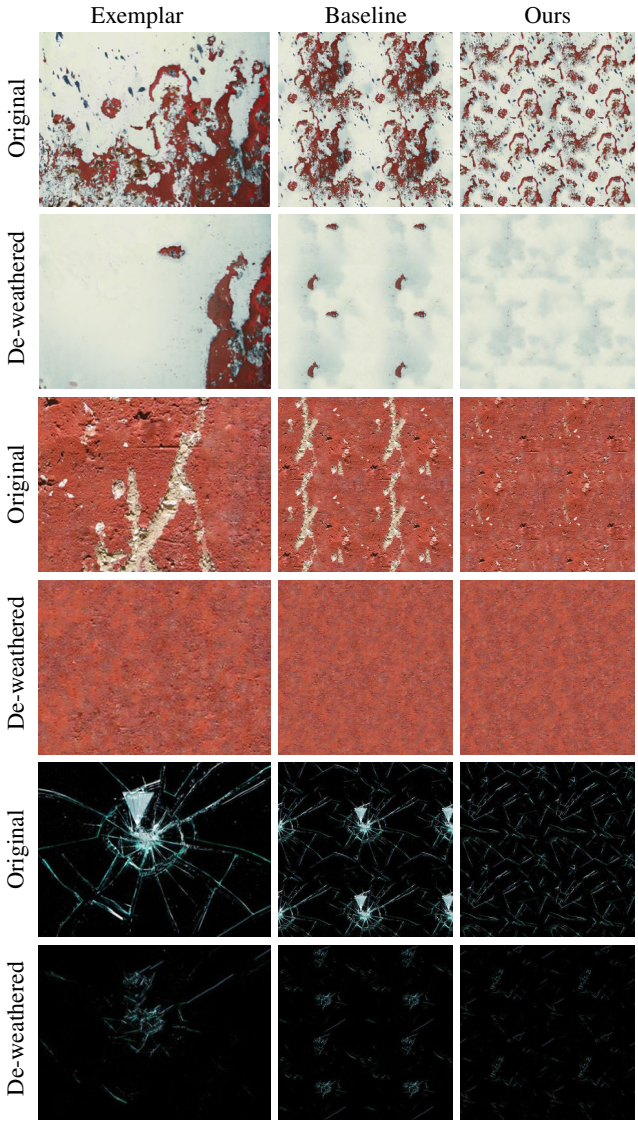

Figure 6: *Texture synthesis performed on input and output of [BKCO16]. The left column is taken directly from their paper: an input exemplar, and the de-weathered output below it. The other columns show the results of running toroidal PatchMatch* (middle) *and our method* (right) *on the respective left-most image.*

Human observers generally perceive repetitiveness more easily as a texture becomes smaller. While we do not explicitly take this into account, this effect could be addressed by increasing the size of the output texture (which may require more texture re-use or a larger input exemplar). Alternatively, a higher level of stationarity could be enforced. Optimal parameters for different viewing conditions could be an interesting avenue for future work.

5.5 Applications

Thanks to our continuous formulation, a user-interface for designing tileable textures could can generate a single texture with locally-prescribed stationarity (Fig. 10). Such a texture has little applied value but can provide an effective user interface. Additionally, we

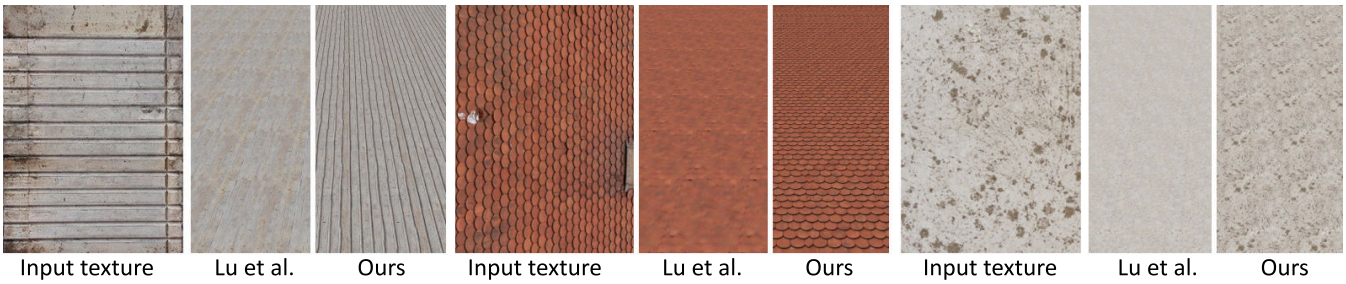

Figure 7: *Comparison between an approach that segments out salient features before synthesizing [LDR09] and our algorithm. In the first two textures, their algorithm shows poor segmentation performance, as critical parts of the structure are masked out. In the third example, our own algorithm starts exhibiting modest repetitiveness, while Lu et al. again excessively remove detail.*

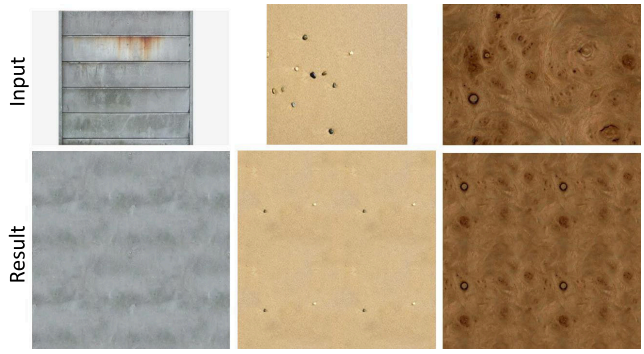

Figure 8: *Three failure cases* (columns) *where our approach could not stationarize the input* (top row) *into a useful result* (bottom row)

compare between a direct gradient and a perceptually uniform one in Fig. 11. Finally, we show a typical mobile-game scene, that has been textured directly from photographs in Fig. 12.

6 Conclusion

Texture mapping, if done right, can increase the visual realism, even on devices with smaller compute capabilities. We have addressed a key limitation of texture mapping in this article: the need of expert knowledge required to turn a texture-like image into a tileable texture that introduces detail, without producing repetition.

Our approach is subject to certain limitations: it assumes the image to be fronto-parallel and we have not yet explored the combination with systems to unproject. An interesting avenue of future research could be to employ stationarity to help unprojection and vice versa. We would think that our methodology could be applied to extensions for BRDFs, 3D textures, video textures or animations that all could account for more explicit stationarity control.

Acknowledgments

This work was funded by UCL's EngD VEIV Centre for Doctoral Training, with sponsoring by the Procter & Gamble Company and kind support by Dr. Paul Matts, and by the Engineering and Physical Sciences Research Council (grant EP/K023578/1).

References

[AAL16] AITTALA M., AILA T., LEHTINEN J.: Reflectance modeling by neural texture synthesis. *ACM Trans. Graph. (Proc. SIGGRAPH) 35*, 4 (2016), 65:1–65:13.

[Att54] ATTNEAVE F.: Some informational aspects of visual perception. *Psy Rev 61*, 3 (1954), 183–193.

[BKCO16] BELLINI R., KLEIMAN Y., COHEN-OR D.: Time-varying weathering in texture space. *ACM Trans. Graph. (Proc. SIGGRAPH) 35*, 4 (2016), 141:1–141:11.

[BN76] BLINN J. F., NEWELL M. E.: Texture and reflection in computer generated images. *Comm. ACM 19*, 10 (1976), 542–547.

[BSFG09] BARNES C., SHECHTMAN E., FINKELSTEIN A., GOLDMAN D.: PatchMatch: a randomized correspondence algorithm for structural image editing. *ACM Tr. Gr. (Proc. SIGGRAPH) 28*, 3 (2009), 24:1–24:11.

[Cae85] CAELLI T.: Three processing characteristics of visual texture segmentation. *Spatial Vision 1*, 1 (1985), 19–30.

[CSHD03] COHEN M. F., SHADE J., HILLER S., DEUSSEN O.: *Wang tiles for image and texture generation*, vol. 22. 2003.

[DRV14] DAI D., RIEMENSCHNEIDER H., VAN GOOL L.: The synthesizability of texture examples. In *Proc. IEEE Conf. Comp. on Vision and Pattern Recognition* (June 2014), pp. 3027–3034.

[DvGNK99] DANA K. J., VAN GINNEKEN B., NAYAR S. K., KOENDERINK J. J.: Reflectance and texture of real-world surfaces. *ACM Trans. Graph. 18*, 1 (Jan. 1999), 1–34.

[EF01] EFROS A. A., FREEMAN W. T.: Image quilting for texture synthesis and transfer. In *Proc. SIGGRAPH* (2001), pp. 341–346.

[EL99] EFROS A. A., LEUNG T. K.: Texture synthesis by non-parametric sampling. In *Proc. IEEE Conf. Comp. on Vision and Pattern Recognition* (1999), vol. 2, pp. 1033–1038.

[ELS08] EISENACHER C., LEFEBVRE S., STAMMINGER M.: Texture synthesis from photographs. *Comp. Graph. Forum (Proc. Eurographics) 27*, 2 (2008), 419–428.

[FA91] FREEMAN W. T., ADELSON E. H.: The design and use of steerable filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence 13*, 9 (1991), 891–906.

[Fue05] FUENTES M.: A formal test for nonstationarity of spatial stochastic processes. *J Multivariate Analysis 96*, 1 (2005), 30–54.

[GEB15] GATYS L. A., ECKER A. S., BETHGE M.: A neural algorithm of artistic style. *arXiv:1508.06576v2* (2015).

[Hae90] HAEBERLI P.: Paint by numbers: Abstract image representations. In *Computer Graphics (Proc. SIGGRAPH)* (1990), vol. 24, pp. 207–214.

[HB95] HEEGER D. J., BERGEN J. R.: Pyramid-based texture analysis/synthesis. In *Proc. SIGGRAPH* (1995), pp. 229–238.

[IKN*98] ITTI L., KOCH C., NIEBUR E., ET AL.: A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence 20*, 11 (1998), 1254–1259.

[JAFF16] JOHNSON J., ALAHI A., FEI-FEI L.: Perceptual losses for real-time style transfer and super-resolution. *arXiv:1603.08155* (2016).

[JK70] JOHNSON N. L., KOTZ S.: *Distributions in statistics. 2: Continuous univariate distributions*. Wiley series in probability and mathematical

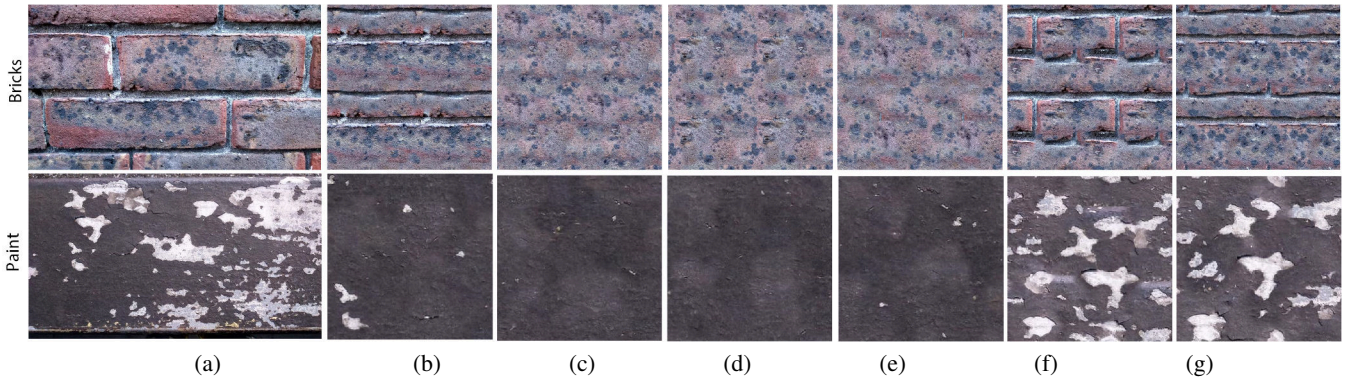

Figure 9: *Comparison of stationarization using different models.* ***(a)*** *The input texture.* ***(b)*** *Stationarization by subtraction of low-pass filter, i.e., matching the local mean.* ***(c–e)*** *Stationarization by fitting 2nd, 3rd and 4th order moments to the local histogram.* ***(f)*** *Using local histogram matching.* ***(g)*** *Using local histogram matching of steerable filter responses. All images show cut-outs, the top row only showing a 2-by-2 tiling.*

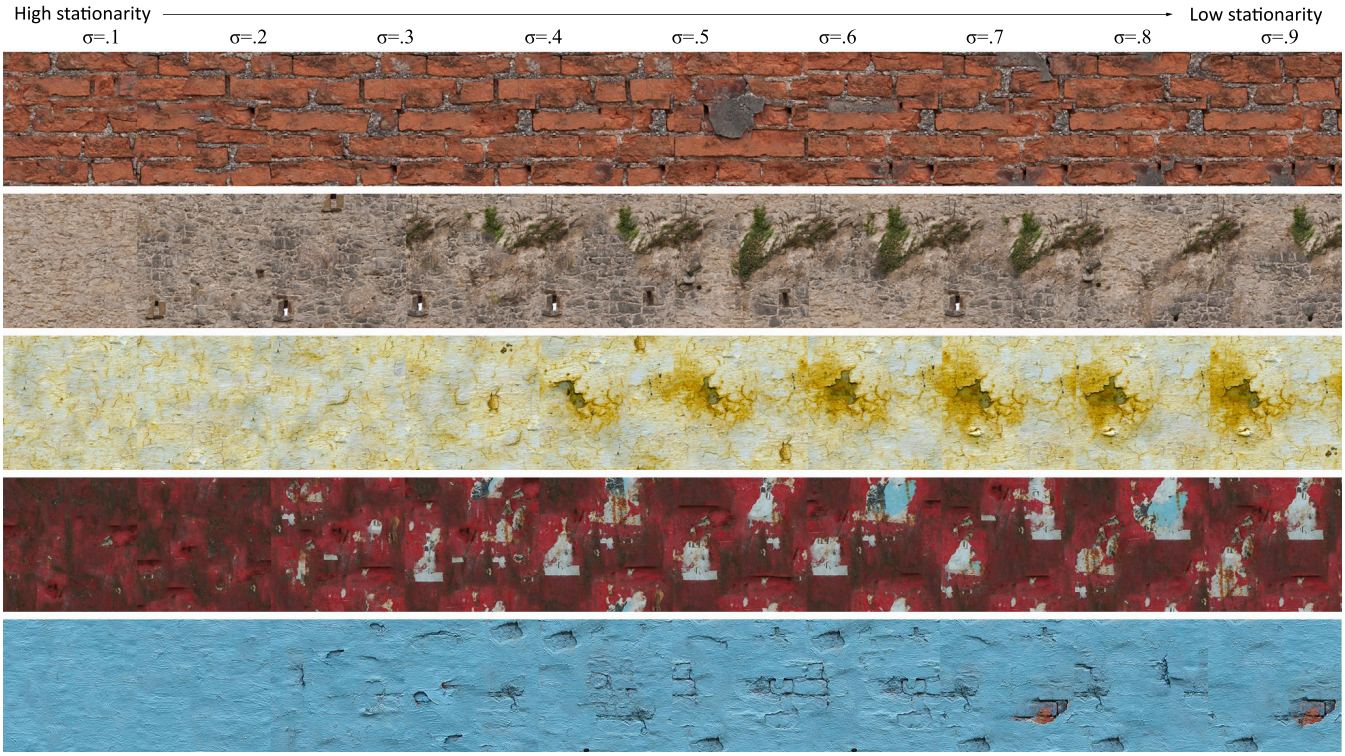

Figure 10: *Gradual variation of stationarity from left to right for different textures in different rows. For each stationarity level we show one texture tile; adjacent tiles often appear near-seamless, due to their common synthesis seed.*

statistics: Applied probability ans statistics. John Wiley & sons, New-York, Chichester, Brisbane, 1970.

[Jul62] JULESZ B.: Visual pattern discrimination. *IRE Trans. Inf Theory 8*, 2 (1962), 84–92.

[Jul65] JULESZ B.: Texture and visual perception. *Scientific American 212*, 2 (1965), 38–48.

[KEBK05] KWATRA V., ESSA I., BOBICK A., KWATRA N.: Texture optimization for example-based synthesis. *ACM Trans. Graph. (Proc. SIGGRAPH) 24*, 3 (2005), 795–802.

[KFCO*07] KOPF J., FU C.-W., COHEN-OR D., DEUSSEN O., LISCHINSKI D., WONG T.-T.: Solid texture synthesis from 2D exemplars. *ACM Trans. Graph. (Proc. SIGGRAPH) 26*, 3 (2007), 2:1–2:9.

[KNL*15] KASPAR A., NEUBERT B., LISCHINSKI D., PAULY M., KOPF J.: Self tuning texture optimization. *Comp. Graph. Forum (Proc. Eurographics) 34*, 2 (2015), 349–359.

[LD06] LAGAE A., DUTRÉ P.: An alternative for Wang tiles: colored edges versus colored corners. *ACM Tr. Graph. 25*, 4 (2006), 1442–1459.

[LDR09] LU J., DORSEY J., RUSHMEIER H.: Dominant texture and diffusion distance manifolds. *Comp. Gr. Forum 28*, 2 (2009), 667–676.

[LH05] LEFEBVRE S., HOPPE H.: Parallel controllable texture synthesis. *ACM Trans. Graph. (Proc. SIGGRAPH) 24*, 3 (2005), 777–786.

[LL12] LASRAM A., LEFEBVRE S.: Parallel patch-based texture synthesis. In *Conf. on High-Perf. Graph.* (2012), pp. 115–124.

[LSA*16] LOCKERMAN Y. D., SAUVAGE B., ALLÈGRE R., DISCHLER J., DORSEY J., RUSHMEIER H.: Multi-scale label-map extraction for

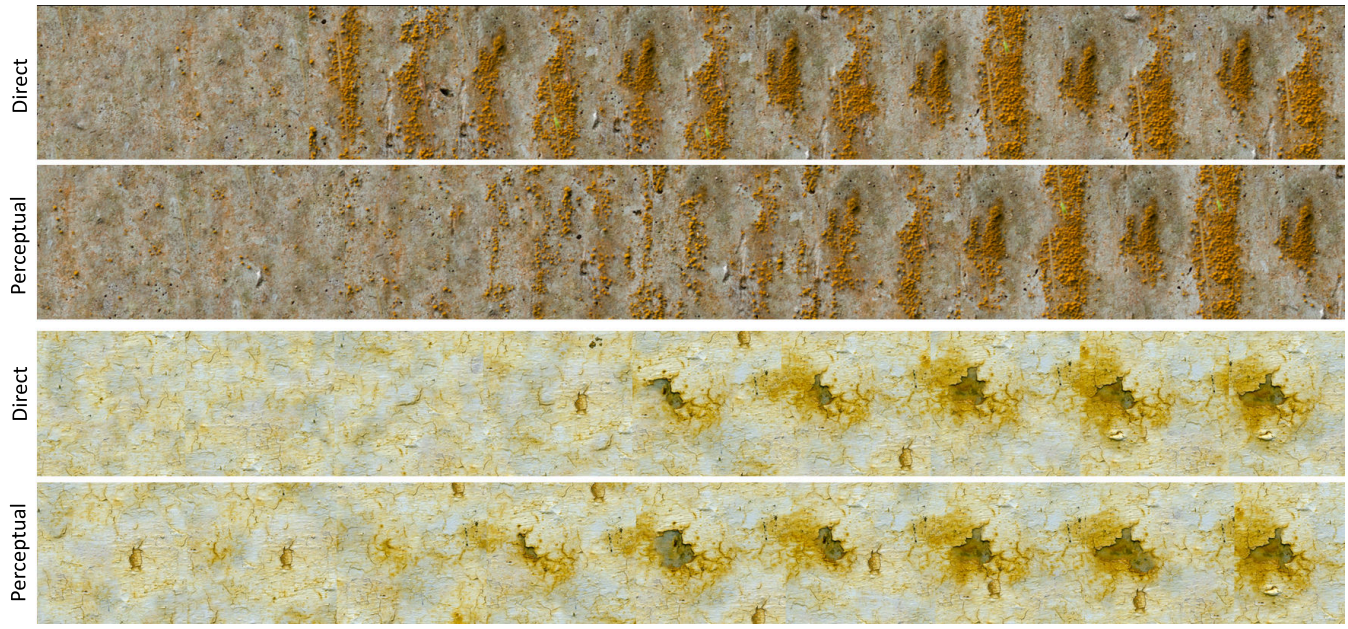

Figure 11: *Comparison of a gradient linear in σ* (1st and 3rd row) *and a perceptually linear variant* (2nd and 4th row). *The linear progression may exhibit abrupt changes of stationarity, while the latter variant spreads across the entire domain allowing for a more convenient navigation.*

Figure 12: *A game scenario showing a variety of objects with tiling textures produced by toroidal PathMatch* (right) *and our approach* (left).

texture synthesis. *ACM Trans. Graph. (Proc. SIGGRAPH) 35*, 4 (2016), 140:1–140:12.

[MZD05] MATUSIK W., ZWICKER M., DURAND F.: Texture design using a simplicial complex of morphable textures. *ACM Trans. Graph. (Proc. SIGGRAPH) 24*, 3 (2005), 787–794.

[Per85] PERLIN K.: An image synthesizer. In *Computer Graphics (Proc. SIGGRAPH)* (1985), pp. 387–396.

[PR69] PRIESTLEY M. B., RAO T. S.: A test for non-stationarity of time-series. *J Royal Stat. Society 31*, 1 (1969), 140–149.

[Ros14] ROSENHOLTZ R.: Texture perception. In *The Oxford handbook of perceptual organization*, Wagemans J., (Ed.). 2014.

[Sim15] SIMONCELLI E. P.: matlabPyrTools. `https://github.com/LabForComputationalVision/matlabPyrTools`, 2015. [Online; accessed 4-January-2017].

[SWRC06] SHOTTON J., WINN J., ROTHER C., CRIMINISI A.: Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In *Proc. European Conf. on Computer Vision* (May 2006).

[TEN14] TAYLOR S. L., ECKLEY I. A., NUNES M. A.: A test of stationarity for textured images. *Technometrics 56*, 3 (2014), 291–301.

[VB78] VICTOR J. D., BRODIE S. E.: Discriminable textures with identical buffon needle statistics. *Bio Cybern 31*, 4 (1978), 231–234.

[VSLD13] VANHOEY K., SAUVAGE B., LARUE F., DISCHLER J.-M.: On-the-fly multi-scale infinite texturing from example. *ACM Trans. Graph. (Proc. SIGGRAPH ASIA) 32*, 6 (Nov. 2013), 208:1–208:10.

[WH13] WANG W., HUA M.: Extracting dominant textures in real time with multi-scale hue-saturation-intensity histograms. *IEEE Trans. Image Processing 22*, 11 (2013), 4237–4248.

[WSI04] WEXLER Y., SHECHTMAN E., IRANI M.: Space-time video completion. In *Proc. IEEE Conf. Comp. on Vision and Pattern Recognition* (2004), vol. 1, pp. 463–476.

[XM10] XU G., MA S.: Evaluation of optimal ω-tile sets for fast texture synthesis. In *Proc. Intl. Conf. Computer Application and System Modeling* (Oct. 2010), vol. 8, pp. V8–342–V8–346.

[YBY*13] YEH Y.-T., BREEDEN K., YANG L., FISHER M., HANRAHAN P.: Synthesis of tiled patterns using factor graphs. *ACM Trans. Graph. 32*, 1 (2013), 3:1–3:13.

[YY14] YANG C.-K., YEH Y.-C.: Stain removal in 2d images with globally varying textures. *Signal, Image and Video Processing 8*, 7 (2014), 1373–1382.

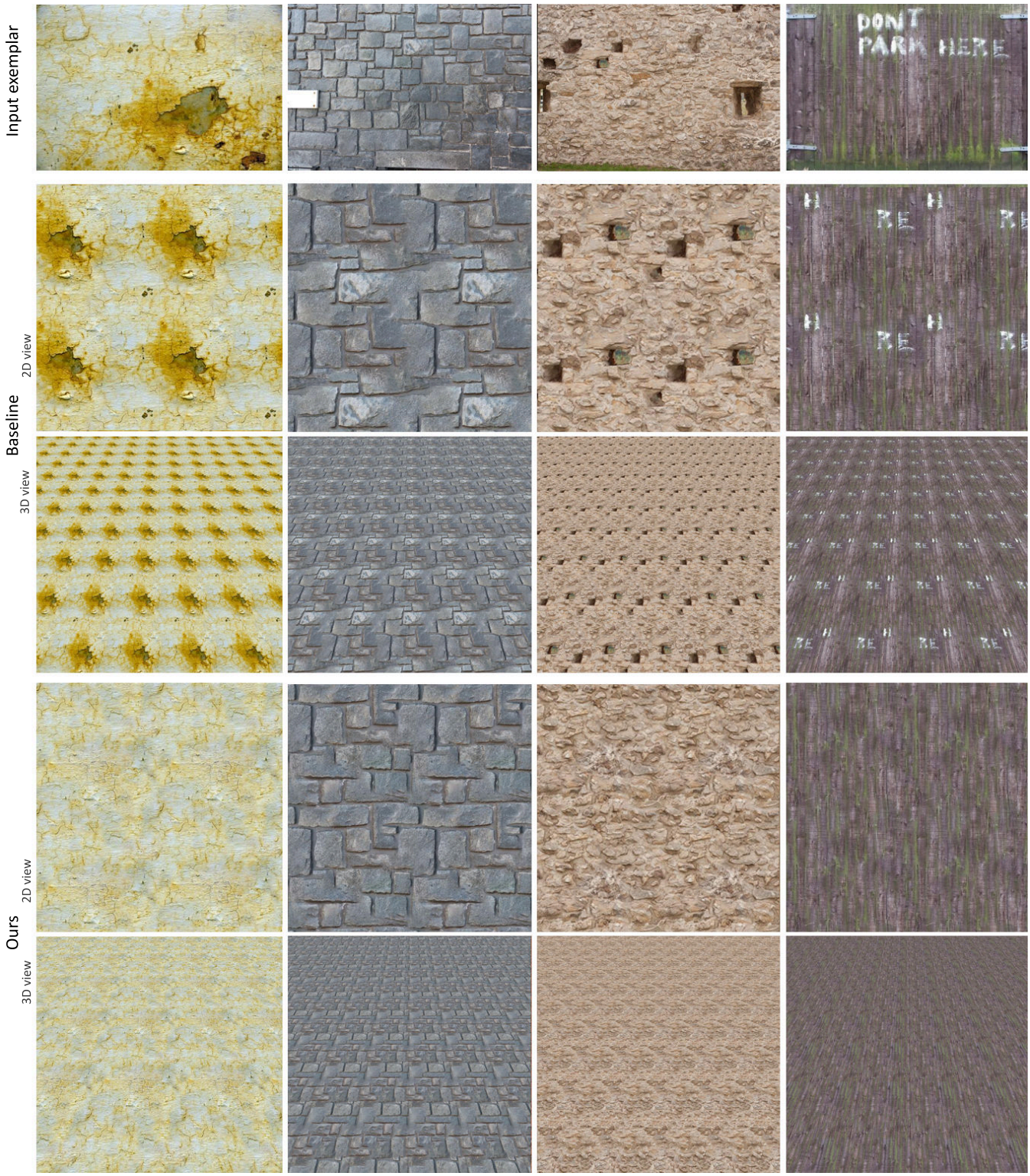

Figure 13: *Starting from different input exemplars* (1st row) *we compare the baseline approach* (2nd and 3rd row) *to our approach* (4th and 5th row). *We show both a 2×2-grid 2D view and a perspective 3D visualization. In all cases, our approach reproduces the exemplars' appearance, yet the results are stationary enough to even tile the large 3D plane without making it appear overly repetitive.*